Un Nouvel Algorithme de Classification Ascendante 2-3 Hiérarchique

A New Agglomerative 2-3 Hierarchical Clustering Algorithm

S. Chelcea[1] P. Bertrand[2,3] B. Trousse[1]

[1] INRIA Sophia-Antipolis, Projet AxIS
[2] ENST-Bretagne, Dept. TAMCIC/IASC, Brest
[3] INRIA Rocquencourt, Projet AxIS

INRIA, Equipe AxIS, BP 93, 06902 Sophia-Antipolis Cedex, France
Sergiu.Chelcea@inria.fr

Résumé

La Classification Ascendante 2-3 Hiérarchique (2-3 CAH) introduite par Bertrand en 2002 [4] [5], généralise la Classification Ascendante Hiérarchique (CAH) en permettant sous certaines conditions, à chaque classe d'être agrégée deux fois durant le processus de formation de classes par agrégations successives. Dans ce papier, nous proposons un nouvel algorithme de la 2-3 CAH qui améliore celui proposé en 2002, par une formulation simplifiée et par une réduction de la complexité de $\Theta(n^3)$ à $\Theta(n^2 \log n)$, où n désigne le nombre d'objets à classer. À l'issue de chaque agrégation, ce nouvel algorithme de la 2-3 CAH permet de supprimer certaines classes, afin que l'ensemble des classes obtenues soit indicé au sens large. Nous avons réalisé la première implémentation d'un algorithme de 2-3 CAH, puis comparé les résultats de la 2-3 CAH (effectuée avec ce nouvel algorithme) avec ceux de la CAH, d'une part sur les données classiques de Ruspini et d'autre part sur des jeux de données simulées uniformément.

Mots Clef

Classification, Hiérarchie, 2-3 Hiérarchie

Abstract

The 2-3 Agglomerative Hierarchical Clustering (2-3 AHC) introduced by Bertrand in 2002 [4] [5], generalizes the Agglomerative Hierarchical Clustering (AHC) by allowing under certain conditions, that each cluster be merged twice during the process of successive mergings. A new algorithm of 2-3 AHC is proposed in this paper. This algorithm improves the one proposed in 2002, by its simpler formulation and by reducing its algorithmic complexity from $\Theta(n^3)$ to $\Theta(n^2 \log n)$, where n denotes the number of objects to be clustered. At the end of each merging step of the new algorithm of 2-3 AHC, some clusters are removed, if necessary, in order that the obtained set of clusters be weakly indexed. We have implemented for the first time a 2-3 AHC algorithm, then compared the new algorithm of 2-3 AHC with the algorithm of AHC, by applying these clustering procedures to the classical Ruspini data and to artificial data sets.

Keywords

Clustering, Hierarchy, 2-3 Hierarchy

1 Motivations

Quel que soit son contexte d'utilisation, chaque méthode de classification génère des classes ayant des formes et/ou un type qui lui sont spécifiques, car les caractéristiques des classes sont induites par le critère de classification optimisé par la méthode. De plus, chaque méthode de classification génère un ensemble de classes qui est organisé selon une structure (partition, hiérarchie, classification pyramidale, . . .) qui lui est propre également. Lorsque l'on applique une méthode de classification afin d'effectuer une indexation de cas [2], on cherche à ce que les classes obtenues soient clairement identifiées, dans le but de pouvoir affecter de nouveaux objets (cas) à chaque classe obtenue. Cette approche en indexation de cas est grandement facilitée si les classes admettent une représentation géométrique homogène, car on peut alors plus facilement décrire chaque classe à l'aide de conditions géométriques du type : proximité à un prototype, appartenance à un hypercube dans l'espace défini par les variables décrivant les données, etc. Dans ce texte, notre point de vue est celui de l'étude de l'applicabilité d'une technique de classification récente, appelée Classification Ascendante 2-3 Hiérarchique (cf. [4]), avec en perspective son application au domaine de l'indexation de cas, et plus particulièrement à une plate-forme à objets de Raisonnement à Partir de Cas pour des systèmes de recommandation pour le Web [10]. La Classification Ascendante 2-3 Hiérarchique (2-3 CAH) a été récemment introduite par Bertrand dans [4] et [5], afin de généraliser la Classification Ascendante Hiérarchique (CAH). La 2-3 CAH offre la

même flexibilité de choix de critères de classification que la CAH. Par ailleurs, la CAH génère des classes deux à deux disjointes ou incluses l'une dans l'autre, alors que la 2-3 CAH donne la possibilité à chaque classe d'intersecter au plus une autre classe lorsque l'intersection obtenue est distincte des deux classes. Cette caractéristique permet à la structure en classes obtenue de mettre en évidence des groupes d'objets qui ont les caractéristiques communes de deux autres (ce qui n'est pas possible avec la CAH). Une raison supplémentaire est qu'il a été prouvé dans [4] que la 2-3 CAH génère des classifications pyramidales (introduites indépendamment par Diday [7], et Fichet [8]), donc des structures en classes qui sont facilement visualisables.
Rappelons à ce sujet que les classifications pyramidales présentent un intérêt applicatif certain chaque fois que l'ensemble des données est constitué d'objets qui sont naturellement ordonnés le long d'un continuum, comme par exemple cela se produit avec l'étude des séquences génomique (cf. [1]).
Dans la section qui suit, nous rappelons les concepts de 2-3 hiérarchie et de la 2-3 CAH [4]. Dans la Section 3, nous partons de l'algorithme de la 2-3 CAH proposé dans [4] pour définir un nouvel algorithme de 2-3 CAH qui propose à l'issue de chaque agrégation de classes, d'effectuer une étape d'*affinage* (étape de suppression de classes). Avant de conclure en Section 6, la Section 4 présente une analyse de complexité du nouvel algorithme et la Section 5 présente des tests sur les données de Ruspini, et sur des jeux de données simulées uniformément.

2 2-3 Hiérarchies et algorithme de la 2-3 CAH

Dans ce qui suit, E désigne un ensemble de n objets à classer que nous supposons décrit par une *dissimilarité*, notée δ. Nous considérons des collections de sous-ensembles non vides de E, appelés *classes*. Le lecteur pourra se reporter à un ouvrage général comme [9], en ce qui concerne les définitions usuelles en classification.

2.1 2-3 Hiérarchies

Dans ce texte, les collections de classes examinées sont supposées *ordonnées* selon l'ordre naturel d'inclusion entre parties de E. Nous commençons par généraliser à ce contexte des notions usuelles en classification hiérarchique, comme par exemple celles de successeur/prédécesseur et d'indiçage.
Étant donné une collection de classes, notée $\mathcal{C}$, et un élément X de $\mathcal{C}$, chaque classe qui est un élément maximal de la collection $\{Y \in \mathcal{C} : Y \subset X\}$, est appelée *successeur* de X. Si Y est un successeur de X, alors on dit que X est un *prédécesseur* de Y. Si $X, Y \in \mathcal{C}$ sont tels que $X \cap Y \neq \emptyset$, $X \not\subseteq Y$ et $Y \not\subseteq X$, alors on dit que X *intersecte proprement* Y. Rappelons que $\mathcal{C}$ est appelée *hiérarchie sur* E si $\mathcal{C}$ contient E et ses singletons, et s'il n'existe pas d'intersection propre entre deux classes. La collection $\mathcal{C}$ est dite *indicée au sens large* par une fonction $f : \mathcal{C} \to \mathbf{R}^+$ si f est isotone (i.e. $X \subset Y$ implique $f(X) \leq f(Y)$) et si de plus, la condition $f(X) = f(Y)$ avec $X \subset Y$, entraine que la classe X est égale à l'intersection de ses prédécesseurs. Nous rappelons également que $\mathcal{C}$ est dite *indicée* par f si $X \subset Y$ implique $f(X) < f(Y)$. Dans [4], Bertrand a défini une 2-3 *hiérarchie* sur E comme étant toute collection $\mathcal{C}$ de classes qui contient E et ses singletons, qui est stable par intersections non vides, et telle que chaque classe intersecte proprement au plus une autre classe. Une 2-3 hiérarchie sur E est une famille d'intervalles d'au moins un ordre linéaire (i.e. total) défini sur E, et cette famille possède au plus $\lfloor \frac{3}{2}(\mid E \mid -1) \rfloor$ éléments, en excluant les singletons de E (cf. [4]). De cette propriété, il résulte que chaque 2-3 hiérarchie est une classification pyramidale parcimonieuse [5] (voir la Figure 1 pour un exemple de représentation d'une 2-3 hiérarchie).

Pour terminer, nous justifions la terminologie "2-3 hiérarchie" qui est employée. En fait le terme "2-3 hiérarchie" précise de quelle manière l'ensemble des 2-3 hiérarchies constitue une extension de l'ensemble des hiérarchies - en effet, des définitions il résulte clairement qu'une hiérarchie est un cas particulier de 2-3 hiérarchie. Tout d'abord, nous dirons qu'une paire de classes est *hiérarchique* si ces deux classes ne s'intersectent pas proprement. Soit alors une collection $\mathcal{C}$ de classes qui contient E et ses singletons. Il est clair que $\mathcal{C}$ est une hiérarchie si et seulement si toutes les paires de classes sont hiérarchiques. Si l'on suppose de plus que $\mathcal{C}$ est stable par intersection non vide, alors il est facile de montrer que $\mathcal{C}$ est une 2-3 hiérarchie si et seulement si quel que soit le triplet de classes considéré, 2 au moins des 3 paires de classes possibles sont hiérarchiques, ce qui justifie le terme de 2-3 hiérarchie.

2.2 De la CAH à la 2-3 CAH

Rappelons que le principe de l'algorithme de la CAH consiste, en partant de l'ensemble de classes coincidant avec celui des singletons, à agréger successivement deux classes jusqu'à ce que la classe E soit formée, chaque classe ne pouvant être agrégée qu'une seule fois. De plus, l'algorithme ne réunit deux classes que si elles sont les plus proches - au sens d'un *lien d'agrégation* choisi au préalable, que nous notons μ et que nous appelons simplement "*lien*" dans ce texte. Les liens les plus connus et les plus utilisés sont entre autres le *lien minimum*, le *lien maximum* et le *critère de Ward* (cf. [9]). Quand deux classes X et Y sont regroupées, leur lien qui est noté $\mu(X, Y)$, s'interprète comme une mesure du degré d'hétérogénéité de la classe $Z = X \cup Y$ ainsi formée, et se note $f(Z)$. La fonction f ne définissant pas nécessairement une classification indicée, dans le sens défini à la sous-section précédente, une étape d'affinage (élimination de certaines classes) est effectuée à l'issue de la CAH.
Nous rappelons ci-dessous l'algorithme de la 2-3 CAH, proposé en 2002 dans [4] afin de généraliser la CAH, dans lequel $\mathcal{M}_i$ ($i \geq 0$) désigne l'ensemble des classes candidates lors de l'itération $i + 1$.

Algorithme de la 2-3 CAH.

1. **Initialisation.** Poser $i = 0$ et définir $\mathcal{M}_0$ comme étant l'ensemble des singletons de E.
2. **Agrégation.** Poser $i = i + 1$; réunir deux classes X_i et Y_i telles que $\mu(X_i, Y_i) \leq \mu(X, Y)$ pour toute paire $\{X, Y\} \subseteq \mathcal{M}_{i-1}$ telle que X et Y sont incomparables et vérifient (α) ou (β):
   - (α) X et Y sont maximales, et si une classe intersecte proprement Y (resp. X), alors cette classe est nécessairement la classe X (resp. Y).
   - (β) L'une des classes X ou Y est maximale, et l'autre admet un prédécesseur unique Z. Aucune classe n'intersecte proprement X, Y ou Z.
3. **Mise à jour.** Mettre à jour $\mathcal{M}_i \leftarrow \mathcal{M}_{i-1} \cup \{X_i \cup Y_i\}$: $\mathcal{M}_i$ est privé de toute classe strictement incluse dans au moins une classe de $\mathcal{M}_{i-1}$ et dans $X_i \cup Y_i$.
   Mettre à jour μ en utilisant une extension de la formule de Lance et Williams.
   Mettre à jour f en utilisant :
   $f(X_i \cup Y_i) = \max\{f(X_i), f(Y_i), \mu(X_i, Y_i)\}$.
4. **Test d'arrêt.** Répéter 2 et 3, jusqu'à ce que la classe E soit créée.
5. **Affinage.** Supprimer des classes pour que f définisse un indiçage au sens large.

Cet algorithme utilise la formule de Lance et Williams généralisée (cf. [3]) qui permet la mise à jour de chaque valeur de μ en $\Theta(1)$ pour la plupart des liens classiques (sauf le lien du saut minimum). Il a été établi dans [4], que cet algorithme converge pour tout lien μ, en au plus $\Theta(n^3)$ opérations élémentaires, qu'après chaque étape 3 l'ensemble des classes créées est une 2-3 hiérarchie à la condition près que E soit une classe, et que de plus, la 2-3 hiérarchie obtenue à l'issue de l'étape d'affinage est indicée au sens large par f.

3 Nouvel algorithme de la 2-3 CAH

Nous présentons ici un nouvel algorithme de la 2-3 CAH (cf. Section 3.4), dérivé du précédent, et dont l'intérêt est double : d'une part, son principe se rapproche de celui de l'algorithme classique de la CAH grâce à l'ajout d'une étape d'agrégation intermédiaire (cf. Section 3.2), et d'autre part, nous verrons que la formulation simplifiée de la mise à jour de $\mathcal{M}_i$ (cf. Section 3.1 et sa proposition) et l'intégration de la phase d'affinage à l'intérieur de la phase d'agrégation (cf. Section 3.3), conduit à une réduction de la complexité de l'algorithme (cf. Section 4).

3.1 Reformulation de la mise à jour de l'ensemble $\mathcal{M}_i$ de candidats

Nous commençons par reformuler la procédure de mise à jour de l'ensemble $\mathcal{M}_i$ des classes candidates (étape 3 de l'itération $i + 1$). Le but est de réduire la complexité de cette procédure de $\Theta(n^2)$ à $\Theta(n)$ dans le pire des cas[1], en précisant exactement les classes à supprimer.

1. Dans la suite, toutes les mesures de complexité effectuées seront relatives au nombre maximum d'opérations nécessaires.

Proposition 3.1 *Dans l'algorithme de la 2-3 CAH, on peut sans changer les résultats d'agrégation, choisir $\mathcal{M}_i$ (Étape 3) de la façon suivante : $\mathcal{M}_i$ est égal à $\mathcal{M}_{i-1}$ augmenté de $X_i \cup Y_i$ et privé des classes suivantes :*

- successeurs de X_i et les successeurs de Y_i;
- classes X_i et Y_i, lorsque $X_i \cap Y_i \neq \emptyset$ ou lorsque l'agrégation de X_i avec Y_i est de type (β).

Preuve. Dans l'algorithme initial, comme dans la nouvelle formulation, $\mathcal{M}_i$ est égal à $\mathcal{M}_{i-1}$ augmenté de $X_i \cup Y_i$ et privé de certaines classes incluses dans $X_i \cup Y_i$. Pour comparer les deux approches, il suffit donc d'examiner seulement les classes de $\mathcal{M}_{i-1}$ qui sont incluses dans $X_i \cup Y_i$. Examinons tout d'abord les successeurs de X_i ou de Y_i. Dans l'algorithme initial ils n'appartiennent pas à $\mathcal{M}_i$, car ils sont inclus dans X_i ou Y_i, et dans $X_i \cup Y_i$. C'est également le cas dans la nouvelle formulation. Par ailleurs, dans les deux manières de choisir $\mathcal{M}_i$, si une classe W est incluse dans un des successeurs de X_i (resp. Y_i), alors W n'appartient pas à $\mathcal{M}_{i-1}$, car W a nécessairement déjà été éliminée de $\mathcal{M}_{i'}$ pour un certain $i' \leq i-1$. Par ailleurs, X_i et Y_i sont les seuls successeurs de $X_i \cup Y_i$, donc X_i et Y_i sont les seules classes qui restent à examiner. A priori, trois cas seulement sont envisageables selon que l'agrégation de X_i et Y_i, est de type (α) avec $X_i \cap Y_i = \emptyset$ (`Cas (a)`), ou de type (α) avec $X_i \cap Y_i \neq \emptyset$ (`Cas (b)`), ou de type (β) (`Cas (c)`).

`Cas (a)` - *Agrégation (α) avec $X_i \cap Y_i = \emptyset$.* Dans ce cas, $X_i \cup Y_i$ est la seule classe contenant X_i (resp. Y_i), car X_i (resp. Y_i) était maximale avant la formation de $X_i \cup Y_i$. Ni X_i ni Y_i ne sont retirées de $\mathcal{M}_i$ dans l'algorithme initial, et c'est aussi le cas dans la nouvelle formulation. Donc les deux formulations sont ici équivalentes.

`Cas (b)` - *Agrégation (α) avec $X_i \cap Y_i \neq \emptyset$.* Pour la même raison que dans le cas `(a)`, ni X_i ni Y_i ne sont retirées de $\mathcal{M}_i$ dans l'algorithme initial. Par contre, X_i et Y_i n'appartiennent pas à $\mathcal{M}_i$, si l'on utilise la nouvelle formulation. Cependant selon l'algorithme initial, ni X_i ni Y_i ne seront agrégées lors d'une réunion ultérieure de cet algorithme. En effet d'une part, aucune des classes X_i et Y_i ne peut être utilisée pour une agrégation de type β, car X_i et Y_i s'intersectent proprement. Et d'autre part, aucune des classes X_i et Y_i ne peut être utilisée pour une agrégation de type α, car X_i et Y_i ne sont plus maximales.

`Cas (c)` - *Agrégation (β).* Supposons - sans perte de généralité - que Z soit le prédécesseur (unique) de X_i. Donc $X_i \notin \mathcal{M}_i$ dans l'algorithme initial, mais $Y_i \in \mathcal{M}_i$ car Y_i est inclus dans une seule classe de $\mathcal{M}_i$ (i.e., $X_i \cup Y_i$). Par contre, X_i et Y_i n'appartiennent pas à $\mathcal{M}_i$, si l'on utilise la nouvelle formulation. Cependant selon l'algorithme initial, Y_i ne sera pas agrégée lors d'une agrégation ultérieure de l'algorithme. En effet, Y_i possède un prédécesseur unique $X_i \cup Y_i$ mais $X_i \cup Y_i$ intersecte proprement Z (car Z contient strictement X_i mais est disjoint de Y_i). Donc Y_i ne pourra pas être utilisée pour une agrégation quelle soit de type β ou α.

En conclusion, bien que le choix de l'ensemble $\mathcal{M}_i$ diffère dans certains cas selon les deux approches, les classes que l'on peut réunir selon les deux approches sont bien les mêmes. Par conséquent, la nouvelle façon de choisir $\mathcal{M}_i$ ne change pas les possibilités de réunion entre classes à chaque itération. □

3.2 Ajout d'une agrégation de type α à la suite de chaque agrégation de type β

La propriété suivante met en évidence le fait qu'après une agrégation de type β, la réunion de la nouvelle classe $X_i \cup Y_i$ et de l'unique prédécesseur Z de X_i ou de Y_i, est toujours réalisée ultérieurement par l'algorithme.

Proposition 3.2 *Si l'agrégation de la i^{eme} étape est de type β, alors la classe $X_i \cup Y_i$ formée à cette étape, sera nécessairement réunie avec le prédécesseur de X_i ou de Y_i, lors d'une étape ultérieure de l'algorithme.*

Preuve. Supposons - sans perte de généralité - que Z soit le prédécesseur (unique) de X_i, avant l'agrégation β de X_i et Y_i, et de plus, plaçons-nous à la fin de cette étape d'agrégation. Il est clair que $X_i \cup Y_i$ est une classe maximale de $\mathcal{M}_i$. Si Z n'est pas maximale, alors $X_i \subset Z \subset Z'$, ce qui entraine que X_i a été éliminée de $\mathcal{M}_{i'}$ ($i' < i$) au plus tard lors de la mise à jour qui suit la formation de Z', ce qui contredit $X_i \in \mathcal{M}_{i-1}$. Donc Z est maximale, et $Z \in \mathcal{M}_i$, car on ne peut pas éliminer une classe maximale de $\mathcal{M}_j$ avec $j \leq i$. Il en résulte que les classes $X_i \cup Y_i$ et Z appartiennent à $\mathcal{M}_i$, sont maximales et s'intersectent proprement. Donc ces classes peuvent être réunies selon la condition (α). De plus, la classe $X_i \cup Y_i$ (resp. la classe Z) ne peut être réunie qu'à la classe Z (resp. la classe $X_i \cup Y_i$) par définition de l'algorithme. Si ces deux classes ne sont pas réunies, cela signifie que l'on réunit deux classes qui possèdent une valeur de lien inférieure ou égale à $\mu(X_i \cup Y_i, Z)$, et que les deux classes ainsi réunies sont incluses dans $E - (X_i \cup Y_i \cup Z)$, car d'une part aucun successeur de X_i ou de Y_i n'appartient à $\mathcal{M}_j$ ($j \geq i+1$) et d'autre part, aucun successeur de Z n'est agrégeable car Z intersecte proprement la classe $X_i \cup Y_i$. Cependant, l'algorithme ne se terminera pas (et on sait qu'il se termine nécessairement d'après [2]) tant que la classe E n'est pas créée. Or E ne peut pas être créée tant que l'on réunit des classes incluses dans $E - (X_i \cup Y_i \cup Z)$, donc tant que la réunion de $X_i \cup Y_i$ et de Z n'est pas réalisée, ce qui achève la preuve. □

Proposition 3.3 *Si après chaque agrégation de type β, on réunit $X_i \cup Y_i$ (itération i quelconque) avec le prédécesseur Z (de X_i ou de Y_i), alors à la fin de chaque étape 2 ainsi modifiée, aucune classe n'intersectera proprement une classe maximale.*

Preuve. Raisonnons par récurrence sur le numéro d'itération i. Le résultat est vrai pour $i = 1$. Supposons qu'il le soit jusqu'à l'itération $(i-1)$ incluse avec $i > 1$, et considérons l'itération i. Toute itération se terminant par une agrégation de type α, on vérifie aisément que la dernière classe formée est une classe maximale qui n'intersecte proprement aucune autre classe : cela résulte du fait que cette classe peut toujours être considérée comme réunion de deux classes maximales à l'itération précédente, donc par hypothèse de récurrence, n'intersectant proprement aucune autre classe créée au plus tard à l'itération $(i-1)$. D'où le résultat. □

Il résulte de cette proposition que si à la suite de chaque agrégation de type β, on effectue systématiquement une agrégation de type α dans les conditions décrites par la proposition 3.3 (nous dirons dans ce cas que l'agrégation de type α est *intermédiaire*), alors les classes maximales obtenues en fin de chaque itération, forment une partition de E.

Donc l'ajout de l'agrégation intermédiaire lors de l'étape 2 permet de reformuler la condition d'agrégation de l'algorithme initial de la manière suivante : pour tout $\{X, Y\} \subseteq \mathcal{M}_{i-1}$ tel que X et Y sont incomparables, on a (α) : "X et Y sont maximales", ou (β) : "une seule des classes X ou Y est maximale". Par conséquent, deux classes incomparables X et Y de $\mathcal{M}_{i-1}$ peuvent être réunies si et seulement si au moins l'une des deux classes X ou Y est maximale.

3.3 Intégration de l'étape d'affinage dans l'étape d'agrégation

Nous proposons d'intégrer l'étape d'affinage dans l'étape d'agrégation. Cette intégration permet d'obtenir un indiçage au sens large par f sans recourir à l'étape 5 d'affinage. Plus précisément, chaque fois qu'une classe $X \cup Y$ est créée, nous comparons $f(X \cup Y)$ avec $f(X)$ et $f(Y)$. Si $f(X \cup Y) = f(X)$ (resp. $f(X \cup Y) = f(Y)$), nous éliminons X (resp. Y), à condition que $X \cup Y$ soit le seul prédécesseur de X (resp. Y). Ce cas est illustré dans l'exemple de la Figure 1, où $f(X) < f(Y) = f(X \cup Y)$: Y doit donc être supprimé de la structure.

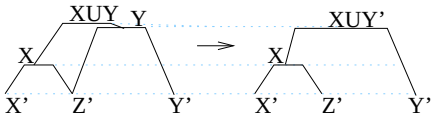

FIG. 1 – *Exemple d'affinage intégré*

3.4 Proposition d'un nouvel algorithme de la 2-3 CAH

Nous décrivons ci-dessous l'algorithme de la 2-3 CAH obtenu (voir [6] pour plus de détails) en tenant compte des deux changements précédents, à savoir l'ajout d'une agrégation intermédiaire (présenté en 3.2) et l'intégration de la phase d'affinage dans la boucle d'agrégation (présenté en 3.3).

1. **Initialisation.** Poser $i = 0$ et définir $\mathcal{M}_0$ comme étant l'ensemble des singletons de E.
2. a) **Agrégation.** Poser $i = i + 1$; réunir deux classes X_i et Y_i telles que $\mu(X_i, Y_i) \leq \mu(X, Y)$ pour toute paire $\{X, Y\} \subseteq \mathcal{M}_{i-1}$ de classes qui sont incomparables et telles que l'une d'entre elles au moins est maximale. Calcul de $f(X_i \cup Y_i)$.
   b) **Agrégation intermédiaire.** Si Z est prédécesseur de X_i ou de Y_i, et que $Z \neq X_i \cup Y_i$, alors **réunir** Z et $X_i \cup Y_i$, et retirer de $\mathcal{M}_i$ ces deux classes et leurs successeurs. Calcul de $f(X_i \cup Y_i \cup Z)$.
   c) **Mise à jour de** μ : ajout des liens entre la nouvelle classe maximale et les autres classes candidates en utilisant la formule de Lance et Williams généralisée.
3. **Affinage.** Éliminer chaque classe $W \in \{X_i, Y_i, X_i \cup Y_i, Z\}$ admettant un prédécesseur W' tel que : $f(W) = f(W')$.
4. **Mise à jour.** $\mathcal{M}_i$ est mis à jour en ajoutant la dernière classe créée et en éliminant les successeurs des classes qui ont été agrégées, ainsi que les classes agrégées elles-mêmes lorsque leur intersection est propre. Mise à jour de μ, en éliminant les paires non candidates.
5. **Test d'arrêt.** Répéter 2, 3 et 4, jusqu'à ce que la classe E soit créée.

Les classes produites par l'étape 2 forment une 2-3 hiérarchie (cf [4] et Proposition 3.2). L'intégration de l'étape d'affinage (étape 3) à l'intérieur de la boucle définie par les étapes 2-5, assure que la structure classificatoire est indicée au sens large par f. Par ailleurs, il est clair que la suppression de classes ayant un seul prédécesseur (étape 3), ne change pas la propriété que les classes générées par l'algorithme forment une 2-3 hiérarchie.
Enfin, pour que l'algorithme de la 2-3 CAH puisse être effectivement implémenté, il est nécessaire tout d'abord de choisir le lien d'agrégation (lien du saut minimum, lien du saut maximum, etc). Le choix d'un lien d'agrégation en CAH suppose que les classes comparées soient disjointes ou incluses l'une dans l'autre. Comme cette condition n'est pas nécessairement remplie durant les étapes d'agrégation de l'algorithme de la 2-3 CAH, il est nécessaire d'adapter cette définition au cas de deux classes s'intersectant proprement. Par la suite, lors de l'implémentation et des tests, nous utiliserons le lien du *saut minimum* défini par $\mu(X, Y) = \min\{\delta(x, y) : x \in X - Y, y \in Y - X\}$, et le lien du *saut maximum* défini par $\mu(X, Y) = \max\{\delta(x, y) : x \in X, y \in Y\}$ (formules qui permettent de se ramener au cas de deux ensembles disjoints).

3.5 Implémentation

Nous avons réalisé une première conception et implémentation orientée objet d'un algorithme de 2-3 CAH (en Java). Le code a aussi été intégré dans CBR*Tools [10], une plate-forme de Raisonnement à Partir de Cas développée à l'INRIA.

4 Analyse de complexité de l'algorithme

Pour déterminer la complexité de notre algorithme, nous décrivons dans le première paragraphe la structure des données utilisée. Ensuite le deuxième paragraphe réalise l'analyse théorique de la complexité en temps de notre nouvel algorithme de la 2-3 CAH, qui est dans le pire des cas en $\Theta(n^2 \log n)$. Cette analyse théorique est soutenue dans le dernière paragraphe par une validation expérimentale.

4.1 Structure de données

Les opérations les plus coûteuses en temps, sont le stockage et la manipulation de la matrice contenant les valeurs du lien μ entre les classes. Nous effectuons ces opérations à l'aide d'une structure de données *arborescente et ordonnée* qui met en correspondance les valeurs de μ et les paires de classes candidates. Plus précisément, les deux classes qui sont choisies pour être réunies, forment une paire qui minimise les trois critères suivants : (1) minimalité de la valeur du lien μ, puisque nous recherchons deux classes les plus proches au sens du lien d'agrégation, (2) minimalité du *cardinal*, ce qui signifie que la somme des nombres d'éléments des classes à réunir est minimale, quand la valeur minimale du lien μ est atteinte par plusieurs paires et (3) minimalité du *rang lexicographique* des identificateurs de classes, lorsque les deux premiers critères sont satisfaits par plusieurs paires. Le premier niveau de la structure est donc constitué des valeurs ordonnées du lien, le second niveau, des valeurs de cardinalités ordonnées des paires possédant la même valeur de lien, et enfin le troisième niveau contient les identificateurs rangés selon l'ordre lexicographique.

4.2 Calcul de la complexité

La complexité de l'initialisation (étape 1) est en $\Theta(n^2 \log n)$, donc plus grande que dans [4], mais nous verrons que le nombre d'opérations sur la structure arborescente ordonnée est moins important à chaque itération : en $\Theta(n \log n)$ au lieu de $\Theta(n^2)$.
Dans ce qui suit, nous analysons la complexité des étapes 2-4, qui sont répétées au plus $\lfloor \frac{3}{2}(n-1) \rfloor$ fois, jusqu'à ce que la classe E soit créée.
Dans l'étape d'agrégation (étape 2.a), nous recherchons d'abord la paire qui minimise nos critères, en $\Theta(1)$, puis nous créons la nouvelle classe $X_i \cup Y_i$ également en $\Theta(1)$.
La complexité de l'agrégation intermédiaire (étape 2.b) se calcule de la même manière et est identique. La valeur de f pour chaque nouvelle classe est calculée en $\Theta(1)$, sauf dans le cas d'une agrégation intermédiaire où il faut calculer $\mu(X_i \cup Y_i, Z)$ selon une complexité en $\Theta(n)$ pour se ramener à la comparaison de classes disjointes (comparaison de successeurs de Z avec X_i et Y_i) en utilisant la formule de Lance et Williams généralisée. Enfin, les valeurs du lien μ entre la nouvelle classe formée et les autres classes candidates sont calculées (étape 2.c), chacune en $\Theta(1)$ - cas de classes comparées disjointes, puis insérées dans la matrice,

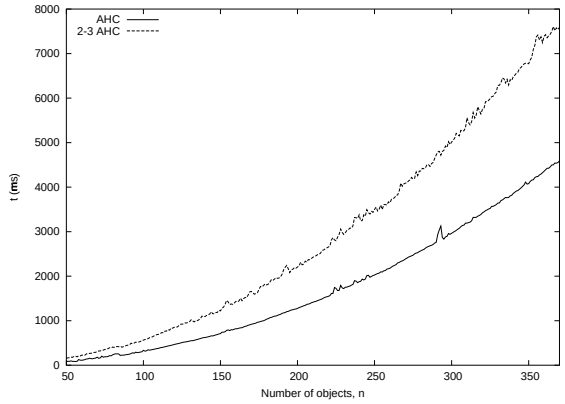

FIG. 2 – *Temps d'exécution*

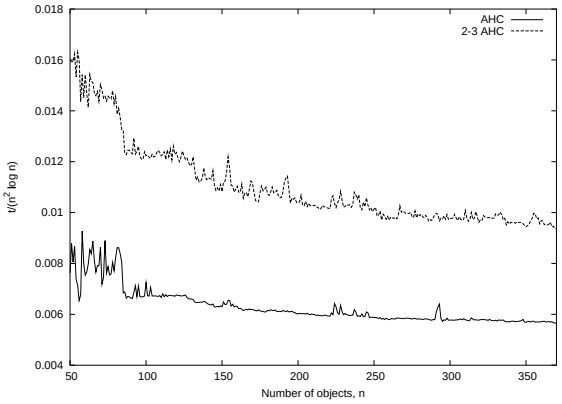

FIG. 3 – *Validation de la complexité*

chacune en $\Theta(\log n)$. Ainsi la complexité totale de l'étape 2 est en $\Theta(n \log n)$.

Dans l'étape d'affinage (étape 3), nous n'éliminons de la structure que les classes qui se trouvent au même niveau que leurs prédécesseurs selon f, et nous mettons à jour les listes *prédécesseurs, successeurs* entre les classes restantes, ce qui s'effectue en $\Theta(n)$, puisqu'une classe peut avoir au plus $\lfloor \frac{3}{2}(n-1) \rfloor$ successeurs.

Dans l'étape de mise à jour (étape 4), nous mettons d'abord à jour $\mathcal{M}_i$ en $\Theta(n)$ puisque l'ajout de la nouvelle classe formée s'effectue en temps constant et puisqu'une classe a au plus n successeurs à éliminer de $\mathcal{M}_i$. Dans la mise à jour de μ, nous éliminons de la structure les paires contenant au moins une classe à éliminer. Puisqu'une paire est éliminée en $\Theta(\log n)$ et que nous avons au plus $\lfloor \frac{3}{2}(n-1) \rfloor$ classes, nous avons ici une complexité en $\Theta(n \log n)$. Par conséquent, la complexité de l'étape 4 est en $\Theta(n \log n)$.

Ainsi, la complexité totale de l'algorithme est $\Theta(n^2 \log n)$ pour l'étape d'initialisation, plus $n \times \Theta(n \log n)$ pour les étapes 2 à 4, soit au total $\Theta(n^2 \log n)$ dans le pire des cas.

4.3 Validation expérimentale de la complexité de l'algorithme

Pour valider l'analyse théorique de complexité de la Section précédente, nous avons appliqué les algorithmes de la CAH et de la 2-3 CAH à des données simulées de façon uniforme à l'intérieur d'un carré (n points aléatoires avec n entre 50 et 400).

La Figure 2 indique le temps d'exécution de ces deux algorithmes, en fonction de la taille n de l'ensemble E des jeux de données simulées uniformément. La Figure 3 montre la convergence du rapport $\frac{Temps\ Execution(n)}{\Theta(n^2 \log n)}$ pour l'algorithme de la 2-3 CAH, ce qui est en accord avec l'analyse théorique de la complexité.

5 Comparaison expérimentale CAH / 2-3 CAH

Ayant déjà clairement établi l'avantage au plan théorique de la version proposée en Section 3 (mêmes classes générées - sans l'étape d'affinage, mais avec une complexité en temps de calcul réduite), nous nous proposons dans ce paragraphe d'appliquer la première implémentation existante de l'algorithme de la 2-3 CAH et de comparer les résultats obtenus à ceux produits par la CAH classique.

Nous avons effectué deux types d'analyse comparative entre la CAH classique et la 2-3 CAH : d'une part sur les données de Ruspini et d'autre part sur des données artificielles.

5.1 Cas des données de Ruspini

Rappelons que les données bien connues de Ruspini [12] contiennent quatre classes avec quelques points intermédiaires (cf. Figure 4). Les classifications obtenues avec la CAH et la 2-3 CAH, en utilisant le lien minimum, conduisent aux observations suivantes. Tout d'abord, la 2-3 hiérarchie (sans affinage) contient nettement plus de classes que la hiérarchie classique: 108 classes obtenues par la 2-3 CAH et 74 par la CAH, soit 46% de classes en plus (ce qui est très proche du pourcentage théorique qui compare le nombre maximal de classes des deux types de structure, i.e. 50%). Il en résulte une meilleure représentation de la structure des données que dans le cas de la CAH (cf. Figure 6), y compris dans le cas de la 2-3 CAH avec affinage, que nous avons représentée dans la Figure 5. De façon plus précise, on peut remarquer que les points 43, 44 de la classe 176 sont plus proches de la classe 175 (points 20-42) que de la classe 153 (points 45, 46, 47), ce qui est bien indiqué par la 2-3 hiérarchie (cf. Figure 5), mais ce qui ne l'est pas par la hiérarchie classique (cf. Figure 6).

5.2 Cas de données artificielles

Tout d'abord, nous avons analysé le nombre de classes créées par les deux méthodes de 2-3 CAH (avec et sans affinage) et celle de CAH. Pour cela, nous avons simulé des jeux de données uniformément à l'intérieur d'un carré (5 itérations pour un n donné entre 50 et 900). En utilisant le lien minimum, nous avons obtenu en moyenne 10.9% de classes en plus (avec un écart type de 1.8) pour l'algorithme de la 2-3 CAH avec l'étape d'affinage et 45.6% de classes de plus (avec un écart type de 0.8) pour l'algorithme sans l'étape d'affinage. Dans le cas du lien maximum, ces

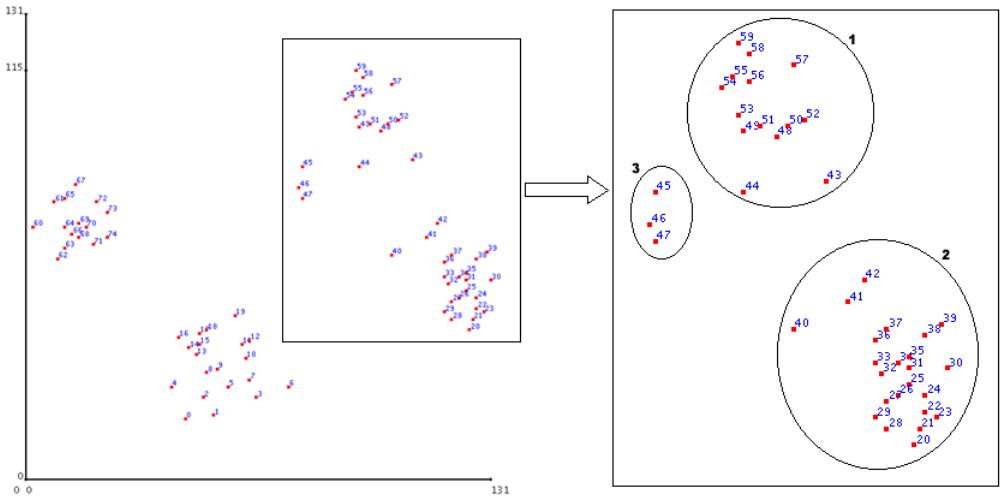

FIG. 4 – *Les données de Ruspini*

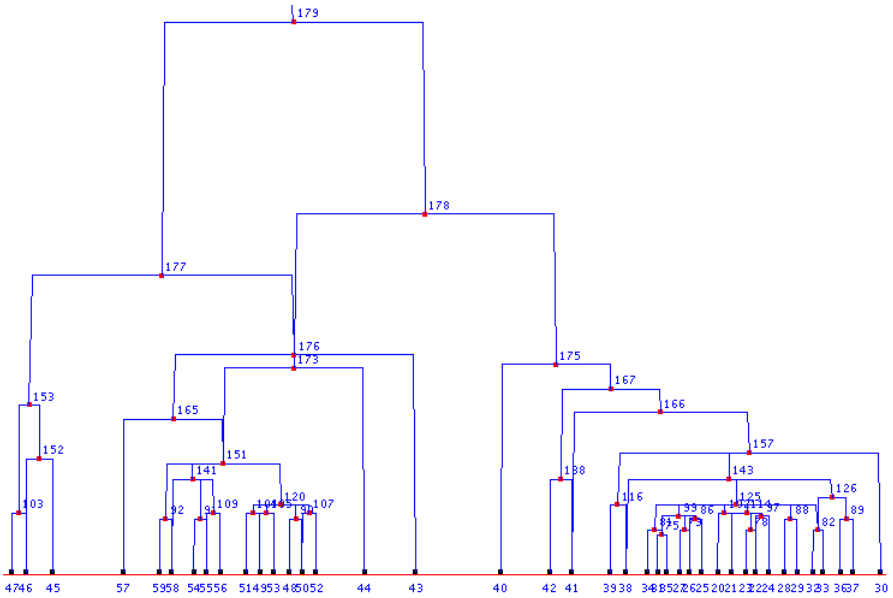

FIG. 5 – *La 2-3 hiérarchie sur les points encadrés de la Figure 4*

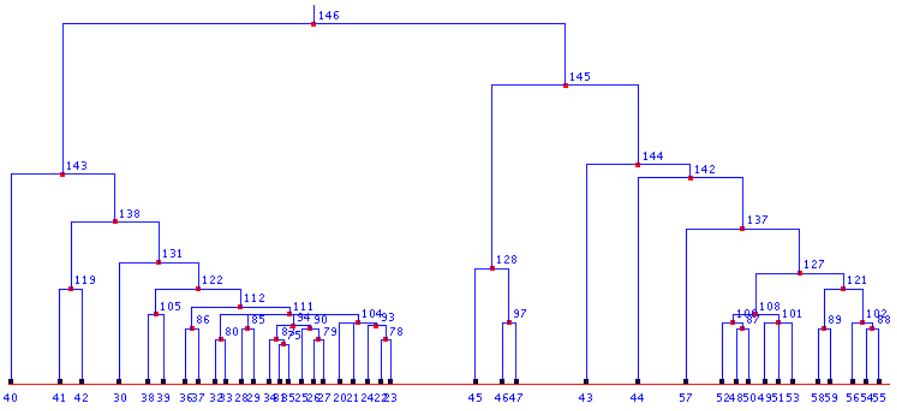

FIG. 6 – *La hiérarchie sur les points encadrés de la Figure 4*

valeurs sont respectivement de 44.6% (1.2) contre 47.5% (0.8) de classes en plus.
Ensuite nous avons simulé uniformément des jeux de données dans le but de s'assurer qu'une structure de données proche de la structure 2-3 hiérarchique, est sinon retrouvée, du moins mieux représentée par une classification 2-3 hiérarchique que par une classification hiérarchique. Comme nous l'avons mentionné plus haut (cf. Section 1), la structure 2-3 hiérarchique est un type de classification pyramidale parcimonieuse qui inclut la classification hiérarchique en tant que cas particulier. Il est donc vraisemblable qu'une classification 2-3 hiérarchique apporte un gain d'information, par rapport au traitement classique hiérarchique, lorsque le jeu de données possède une structure conforme au modèle général de la classification pyramidale - par exemple, des données qui sont ordonnées selon un continuum, comme l'ordre chronologique ou encore l'ordre des lettres d'une séquence génomique. C'est pourquoi, nous avons tout d'abord simulé uniformément n points (25 jeux de données pour chaque $n \in \{10, 20, 30, 40, 50, 75, 100, 125, 150, 175, 200\}$) selon une courbe sinusoidale bruitée par un bruit uniforme : $y_i = sin(2i\pi/n) + u_i$, avec u_i réalisation d'une variable aléatoire uniforme entre 0 et $k \in \{0.02, 0.04, 0.06, 0.08, 0.1\}$.
Nous avons également simulé n points répartis uniformément dans un rectangle de dimensions $[0.5, 50]$. Ces deux types de structure ont en commun la propriété de répartir les données selon un continuum (respectivement, courbe sinusoidale et droite) auquel on ajoute un bruit uniforme, et par conséquent sont bien adaptées à une classification dont les classes sont des intervalles d'un ordre total sur E (ce qui est le cas des hiérarchies et des 2-3 hiérarchies).
Pour analyser l'adéquation des résultats de la CAH classique et de la 2-3 CAH avec ou sans affinage par rapport aux données initiales, nous avons comparé la matrice d des dissimilarités initiales avec la matrice δ des dissimilarités induites qui représente la classification hiérarchique ou 2-3 hiérarchique obtenue : rappelons que $\delta_{i,j}$ est la valeur minimale de $f(X)$ lorsque X est une classe contenant i et j. Pour effectuer cette comparaison, nous avons utilisé la formule du *Stress* [11] : $\Delta = \frac{\sum_i \sum_j [(d_{i,j} - \delta_{i,j})^2]}{\sum_i \sum_j d_{i,j}^2}$.
Le *Stress* mesure le degré de correspondance entre la matrice des dissimilarités reconstruites après l'exécution d'un algorithme (CAH, 2-3 CAH avec ou sans affinage) et celle des dissimilarités initiales.

	Δ_1		Δ_2		Δ_3	
	m.	**e.t.**	**m.**	**e.t.**	**m.**	**e.t.**
sin	0.845	0.019	0.643	0.111	0.649	0.108
rect.	0.969	0.133	0.739	0.167	0.750	0.161

TAB. 1 – *Les moyennes* ***(m.)*** *et les écart types* ***(e.t.)*** *du Stress*

Le Tableau 1 présente les indices de *Stress* pour la CAH classique (Δ_1), la 2-3 CAH avec affinage (Δ_2) et la 2-3 CAH sans affinage (Δ_3).

	G_1 **en %**	G_2 **en %**
sin	23.940	23.132
rect.	23.327	22.163

TAB. 2 – *Les gains de la 2-3 CAH avec ou sans affinage*

Le Tableau 2 contient les gains (en %) apportés par les 2-3 CAH : $G_i = 100(\Delta_1 - \Delta_i)/\Delta_1)$ avec $i = 2, 3$. Nous indiquons ici seulement les valeurs obtenues pour $n = 200$ et $k = 0.06$, les résultats obtenus pour les autres valeurs de n et de k étant globalement similaires (environ 23% de plus avec la 2-3 CAH avec ou sans affinage comparée à la CAH classique)[2].

6 Conclusions et travaux futurs

L'originalité de ce travail est basée sur les quatre points suivants:

(1) un nouvel algorithme de classification de la 2-3 CAH, qui simplifie celui proposé en 2002 (son principe se rapproche de celui de l'algorithme classique de la CAH) ;

(2) une réduction de la complexité de l'algorithme de la 2-3 CAH de $\Theta(n^3)$ à $\Theta(n^2 \log n)$, où n désigne le nombre d'objets à classer ;

(3) une première conception et implémentation orientée objet d'un algorithme de 2-3 CAH (en Java) et son intégration dans CBR*Tools ;

(4) une analyse de la qualité des résultats obtenus avec la 2-3 CAH comparées aux ceux de la CAH classique sur les données de Ruspini et sur des jeux de données simulées uniformément.

Nos travaux futurs et courants concernent les sujets suivants:

(1) poursuite de l'analyse de la qualité de la 2-3 CAH comparée à d'autres méthodes de classification (par exemple les réseaux de neurones) et sur des autres jeux de données ;

(2) étude de l'influence du lien d'agrégation sur les résultats de l'algorithme de la 2-3 CAH ;

(3) étude de l'applicabilité de l'algorithme de la 2-3 CAH dans le contexte d'une application réelle de Fouille de Données, comme le Web Usage Mining.

Références

[1] AUDE J.C., DIAZ-LAZCOZ Y., CODANI J.J, and RISLER J.L. (1999) : *Applications of the Pyramidal Clustering Method to Biological Objects*. Computers & Chemistry 23(3-4): 303-315.

[2] BENEDEK A. and TROUSSE B. (2002) : *Adaptation of Self-organizing Maps for CBR Case Indexing*. Symbolic and Numeric Algorithms for Scientific Computing. 9-12 Octobre, Timisoara, Roumanie.

2. Ces gains ont été obtenus en utilisant le lien maximum.

[3] BERTRAND P. (1986) : *Étude de la Représentation Pyramidale*, Thèse de 3^{ieme} cycle, Univ. Paris-9, France.

[4] BERTRAND P. (2002a) : *Set systems for which each set properly intersects at most one other set - Application to pyramidal clustering*. Cahier du Ceremade numéro 0202, Ceremade, Université Paris-9, France.

[5] BERTRAND P. (2002b) : *Les 2-3 hiérarchies : une structure de classification pyramidale parcimonieuse*. Actes du IX ème Congrès de la Société Francophone de Classification. 16-18 Septembre, Toulouse, France.

[6] CHELCEA S., BERTRAND P. and TROUSSE B. (2002) : *Theoretical study of a new 2-3 hierarchical clustering algorithm*. Symbolic and Numeric Algorithms for Scientific Computing, 9-12 Octobre, Timisoara, Roumanie.

[7] DIDAY E. (1986) : Orders and overlapping clusters in pyramids, in: J. De Leeuw et al., eds., *Multidimensional Data Analysis*, DSWO Press, Leiden, 201-234.

[8] FICHET B. (1987) : Data Analysis: geometric and algebraic structures, in: Y.A. Prohorov and V.V. Sazonov, eds., *Proceedings of the 1st World Congress of the BERNOULLI SOCIETY, Tachkent, 1986*, V.N.U. Science Press, vol. 2, 123-132.

[9] GORDON A.D. (1999) : *Classification*. Chapman and Hall, 2nd ed.

[10] JACZYNSKI M. (1998) : *Modèle et plate-forme à objets pour l'indexation des cas par situations comportementales : application à l'assistance à la navigation sur le Web*, Thèse de doctorat, UNSA, Décembre 1998, France.

[11] JOHNSON A.R. and WICHERN D.W. (1982) : *Applied Multivariate Statistical Analysis*, chapitre 12, Prentince Hall.

[12] RUSPINI E. H. (1969) : *A new approach to clustering*, Inform. control, vol 15, pages 22-32